Efficient RAID Disk Scheduling on Smart Disks

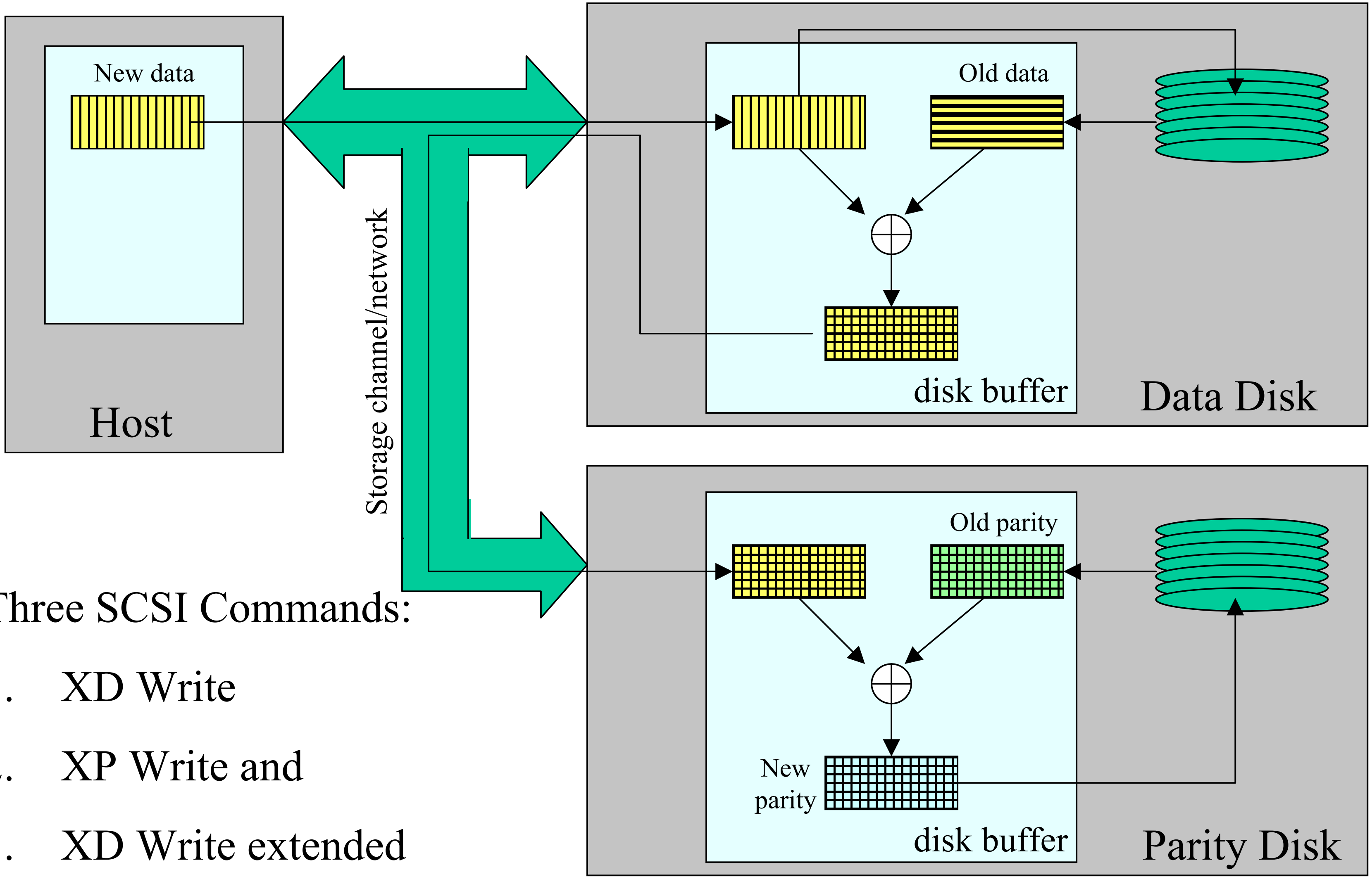
Tai-Sheng Chang & David H.C. Du

Department of Computer Science and Engineering,
University of Minnesota

RAID and Smart Disks

- RAID –Redundant Array of Independent Disks
  - RAID Controllers (cost)
  - Software RAID’s (performance)
- Smart Disks
  - Exclusive-Or (XOR) computation on a disk

Disk-Based XOR



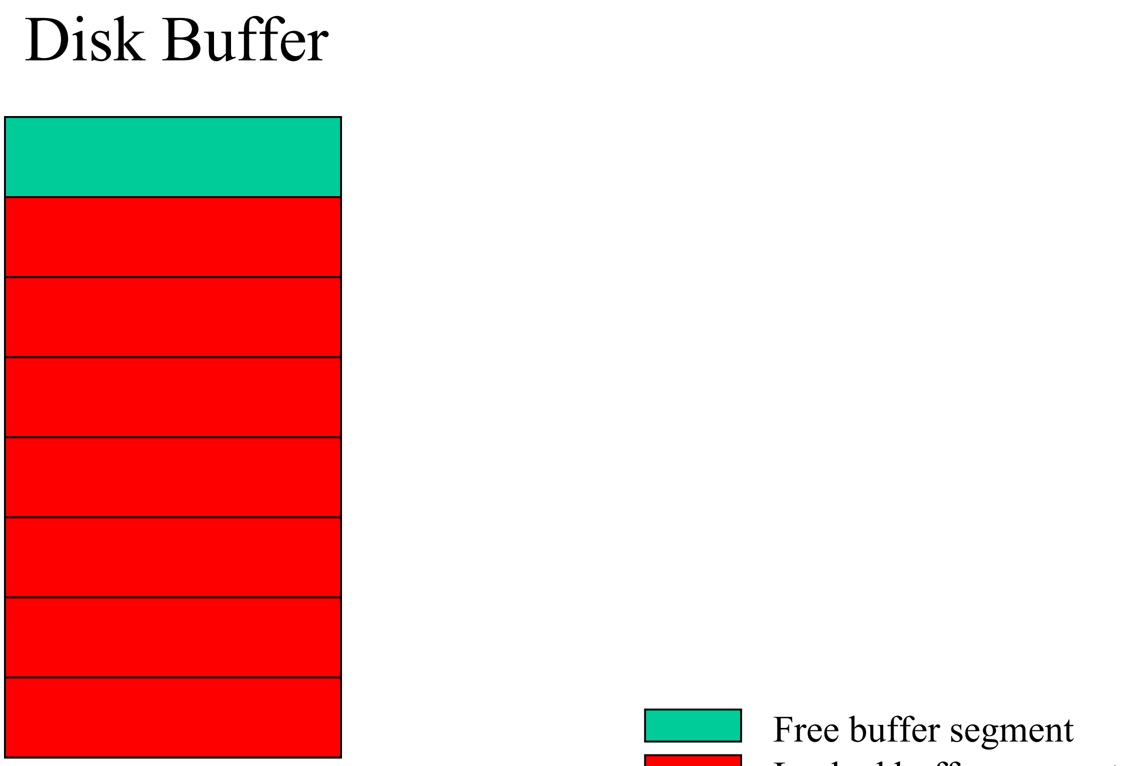
Three SCSI Commands:

1. XD Write
2. XP Write and
3. XD Write extended

RAIDs with Smart Disks

- Opportunities
  - Less data transfer on storage networks (up to 50% reduction)
  - Scalable
  - No RAID controller required
  - CPU load greatly reduced compared to s/w RAIDs
- Challenges
  - Deadlock with single-threaded command executions
  - Out-of-buffer problem with multi-threaded commands
  - Data protection on disks
  - Disk buffer resource requirement
  - Impact on disk efficiency

Disk Buffer Builds Up !



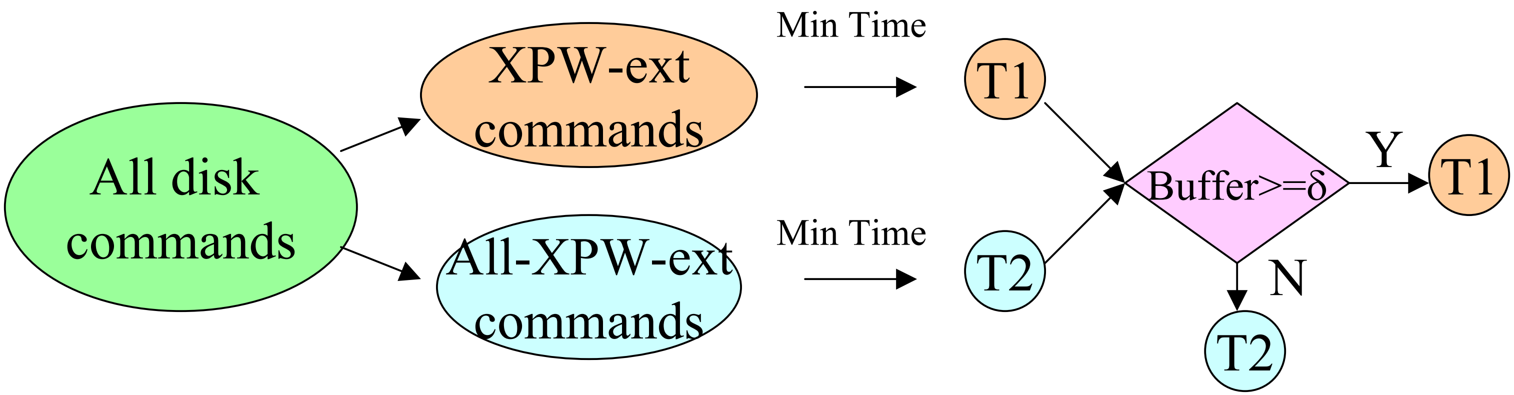
No More XDW-ext can be executed !

How to avoid disk buffer buildup?

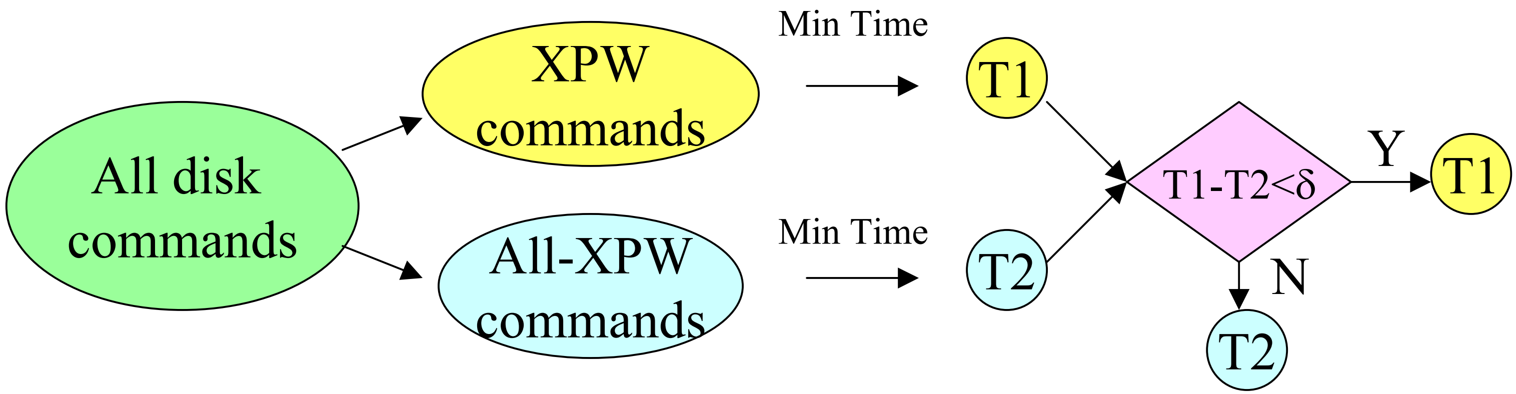
- To avoid buffer running out:
  - Can we slowdown the buffer build-up?
  - Can we free-up the buffer sooner?
- Alternative Approaches
  - Let XPW executed earlier without over-compromising performance -*XPWT*
  - Execute XPW’s when they are too many in disk queue (some other disks are waiting) - *XPWQ*

Scheduling baseline - Greedy Method

- Greedy (shortest time first)
  - Execution order based on disk service time ONLY
  - Exception: When no more buffer space for next XPW-ext, the next non-XPW-ext command in the list will be picked



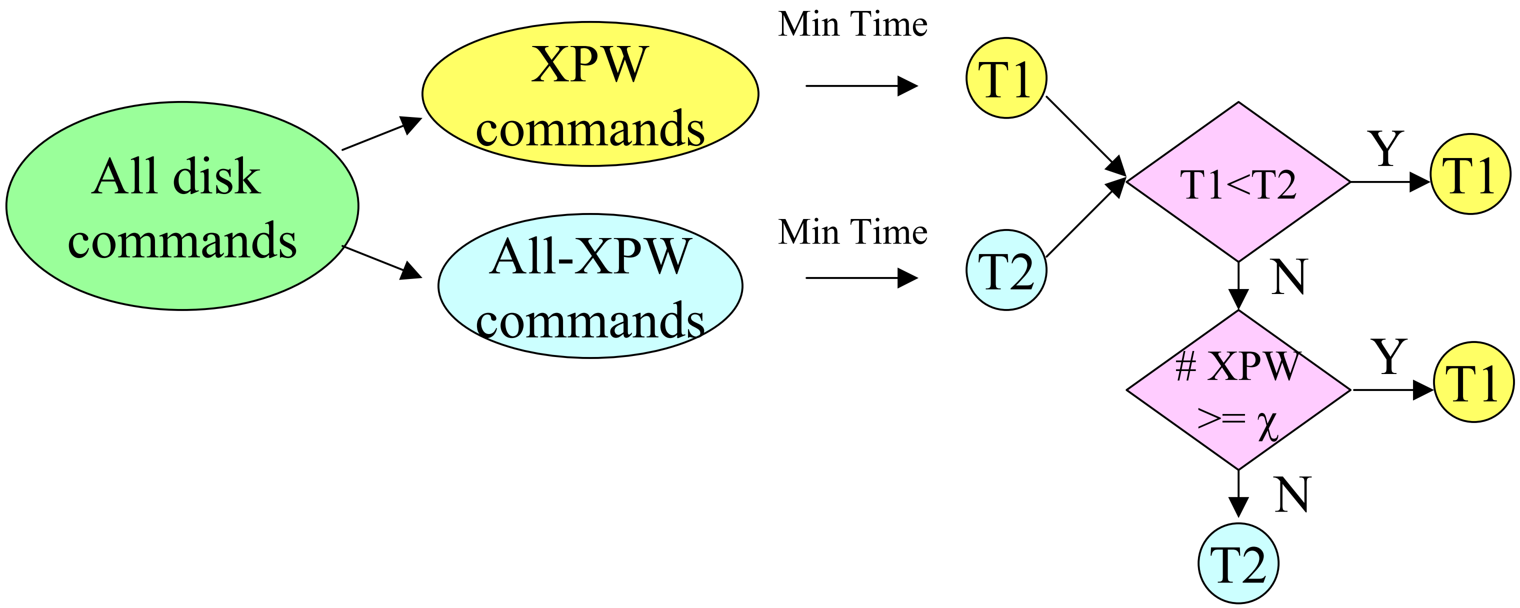
Scheduling Alternative I –XPWT



- Let C_{All}^{min} be the command with the shortest service time T_{All}^{min}
- Let C_{XPW}^{min} be the XPW command with the shortest service time among XPW commands T_{XPW}^{min}.

- If T_{XPW}^{min} - T_{All}^{min} <= δ then choose C_{XPW}^{min} to be the next command.
- Otherwise choose C_{All}^{min}.

Scheduling Alternative II -XPWQ



- Let MaxNxpw be the threshold value of the number of XPW commands.
- Let Nxpw be the number of XPW commands in a disk command queue.

- If Nxpw <= maxNxpw then follow the Greedy Algorithm.
- Otherwise, pick the XPW command with the shortest service time among the those of all XPW's.

Performance Studies –Simulation Model

- Simulation model based on an 8-disk FC-AL model
- Disk, Disk buffer and FC-AL parameters are listed below

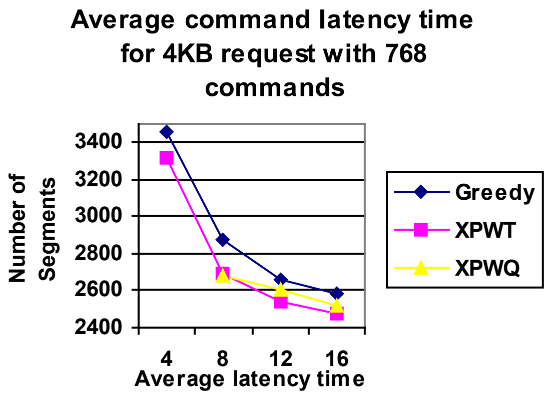
Disk Parameters	Value
Capacity	4.51 GB
Rotation Speed	7202.7 RPM
Average rotation latency	4.17 ms
Seek times	0.5 – 16.5 ms
Transfer rate	5.53 – 7.48 MB/sec

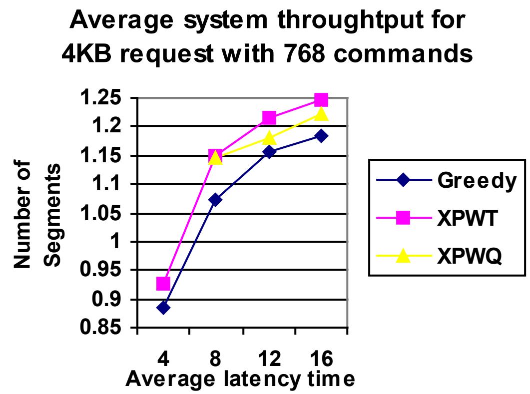
Disk Cache Parameter	Values
Block Size	512 bytes
Number of segments	Varied
Segment size	64 KB

FC-AL Simulation Parameters	Values	Descriptions
Link Speed	100 MB/Sec	Bandwidth of an FC-AL loop
Propagation Delay	3.5 ns	Propagation delay between two nodes
Per Node delay	6 word time	The delay of forwarding a frame by interface
Fairness algorithm	Enabled	The fairness protocol in its arbitration scheme

Simulation Results -1

- Average Latency Time with 768 outstanding 4KB commands

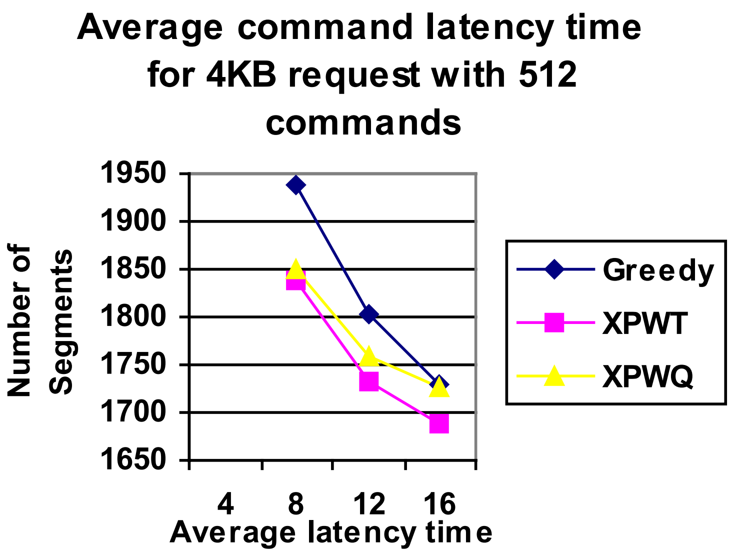


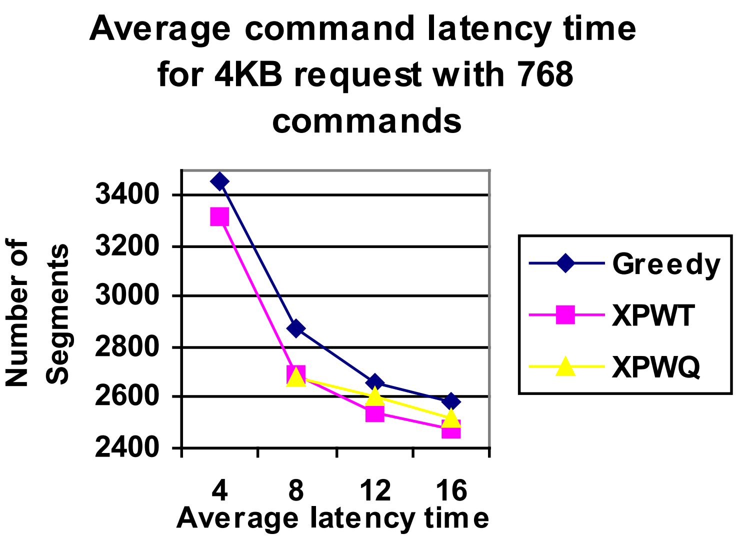


XPWT outperforms the other two. When cache # of the segments =8, it's 8% better

Simulation Results –2

- Average Latency Time with 768 vs. 512 outstanding 4KB commands

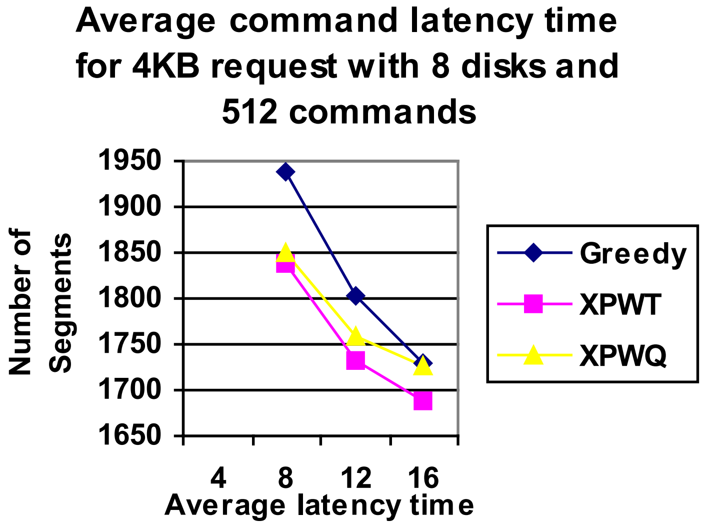


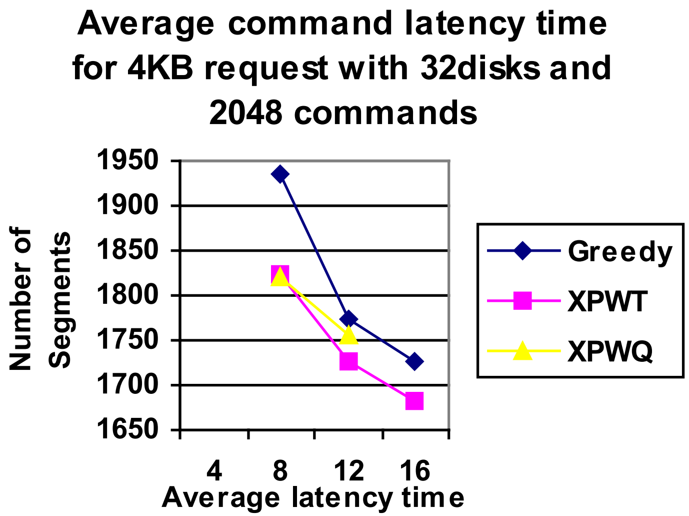


XPWT still performs as much as 5% better than the Greedy in 512 outstanding commands

Simulation Results - 3

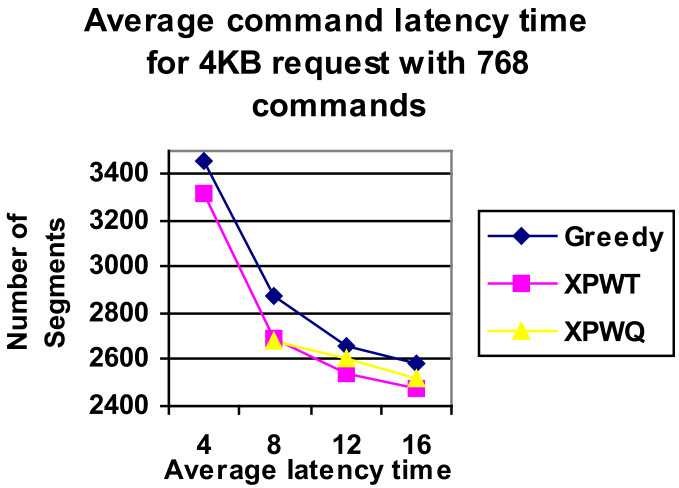
- Average Latency Time with 8 vs. 32 disks

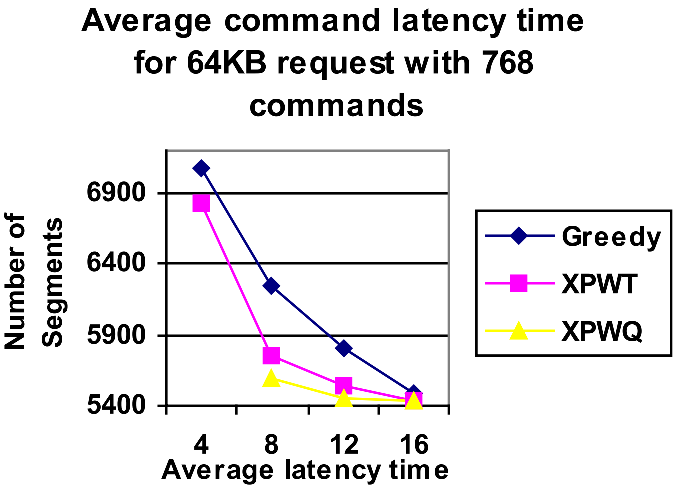




Simulation Results -4

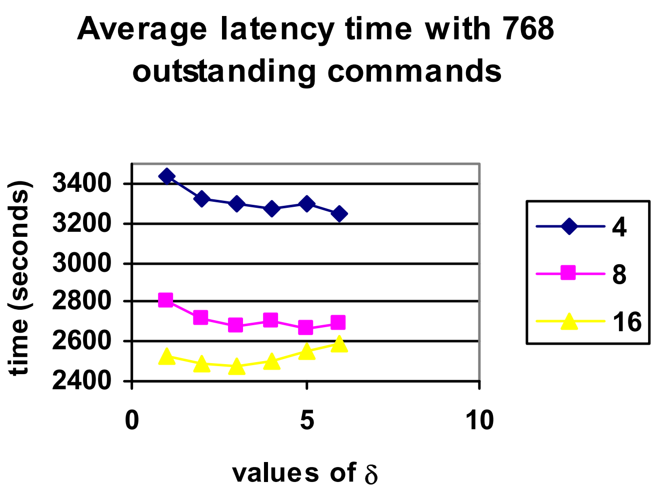
- Average Latency Time with 4KB vs. 64KB commands

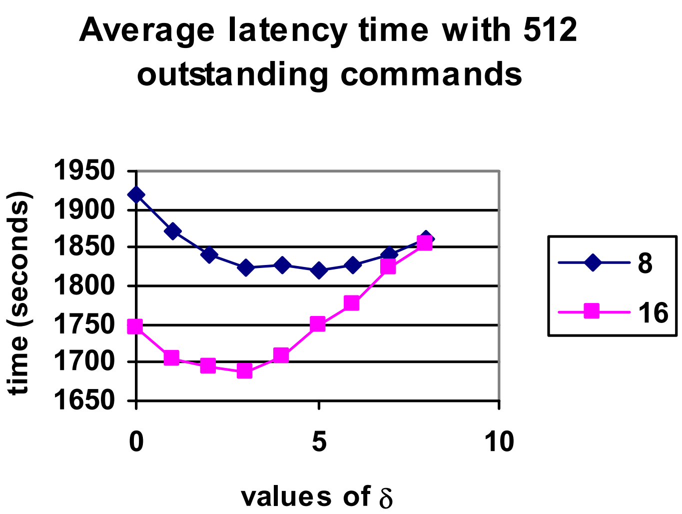




Simulation Results -5

•Average Latency Time with 768 outstanding 4KB commands





Conclusion

- Disk-Based XOR provides a promising low cost alternative to the existing hardware and software RAID solutions
- We have demonstrated both XPWT and XPWQ improved as much as 12% in our test scenarios.
- Rooms for optimization